文字列集合の共通パターンを求める問題について (計算の複雑性に関する研究)

AUTHOR(S):

中津, 楢男; 上林, 弥彦

CITATION:

中津, 楢男 ...[et al]. 文字列集合の共通パターンを求める問題について (計算の複雑性に関する研究). 数理解析研究所講究録 1980, 381: 242-257

ISSUE DATE:

1980-04

URL:

http://hdl.handle.net/2433/104808

数理解析研究所講究録
381巻 1980年 242-257

文字列集合の共通パターンを求める問題について

愛知教育大学　　中津 楢男
京大・工学部　　上林 彌彦

1. まえがき　本稿では、与えられた文字列集合に対して各要素が共通して持つパターンを求める問題について考察する。ここではパターンとして文字列が定義されているアルファベットの要素と変数の連接を考えている。

例えば、{000, 033, 088, 01515}なる集合は0xxという共通パターンを持っている。また、{721327, 331633, 931939}なる集合はxyx^Rというパターンを持っている。この問題はデータ圧縮や集合の特徴を見つける時などに出合う問題である。従来与えられた文字列集合に共通した性質を求める1つの方法として、与えられた文字列集合に対して各要素を含む最小の正規集合を求める問題等が考えられてきた。

また与えられた正系列を受理し、負系列を拒否する状態数最小のオートマトンについても議論されている。

ここでは変数を導入して各要素が共通して持つパターンを

求める問題を考える。この問題は Angluin によって議論されている[1]。

本稿ではまず共通パターンの持つ性質を明らかにし、1変数共通パターンを求めるアルゴリズムを与える。このアルゴリズムは、前処理とデータ構造を工夫することにより従来のアルゴリズムの大巾な改善となっている。2変数以上の共通パターンについては余り良い結果は得られなかった。

2. 基本的定義　Σ を記号の有限集合とする。以下では、Σ 上で定義された文字列を考える。Σ の要素を単に定数と呼ぶことがある。Σ とは異なる記号の有限集合 X ($X \cap \Sigma = \phi$) を考える。X の要素を変数と呼ぶ。

与えられた文字列集合を $S = \{s_1, s_2, \cdots, s_m\}$ (但し, $s_i \in \Sigma^+$, $1 \leq i \leq m$) とする。一般性を失うことなく、各文字列は全て異なり、$|s_1| \leq |s_2| \leq \cdots \leq |s_m|$ であると仮定する。

〔定義1〕　パターン P は $(\Sigma \cup X)^+$ の要素で、長さ有限長のものと定義する。つまり、パターン P は Σ の要素と X の要素を有限個連接したものである。パターン P の長さとは、P の中に現れる定数の数と変数の数の和とする。P の長さを、$|P|$ と書くことにする。

〔定義2〕　パターン P に現れる変数 x を全て、別のパター

ンで置きかえることによって新しいパターンQが得られる時 $P \geq Q$ と書く。但し，Pの中の同じ変数は全て同じパターンで置き換えるものとする。

〔定義3〕 パターンPが生成する文字列集合 $L(P)$ を次の様に定義する。 $L(P) = \{ w \mid w \in \Sigma^+, w \leq P \}$。

与えられた文字列集合Sに対して、$S \subseteq L(P)$ を満たすパターンPをSの共通パターンと呼ぶ。$S \subseteq L(Q) \subset L(P)$ なる2つのパターンP，Qが存在する時，Pは冗長な共通パターンであると言う。また、その様なパターンQが存在しない時，Pを冗長でない共通パターンと呼ぶ。

〔例1〕 $\Sigma = \{0, 1\}$，$P = 0xx$ とした時 $L(P) = \{ w \mid w = 0tt, t \in \{0,1\}^+ \}$ である。また $S = \{1111, 111111\}$ とした時，$1x1x$ は冗長でない共通パターンであるが，xx は冗長な共通パターンである。

3. 1変数共通パターンの性質　本節では与えられた文字列集合の1変数共通パターンの性質について述べる。

〔性質1〕(1) 1変数共通パターンP、Qに対して

$Q \leq P \iff L(Q) \subseteq L(P)$ が成りたつ。

〔性質2〕 冗長でない共通パターンは与えられた文字列集合に対して一意には決らない。一般に相異なる1変数共通パ

ターンの数は、冗長でないものだけを考えても、与えられた文字列の長さに対して指数関数的に増大する場合がある。

〔例2〕 与えられた文字列集合を $\{s_1 = a^{2^k}, s_2 = a^{2^{k+1}}, \cdots, s_m = a^{2^{k+m-1}}\}$ とする。$n = |s_1| = 2^k$ とする。

この時、任意の整数 i $(0 \leq i \leq k)$ について、2^i 個の変数 x と $2^k - 2^i$ 個の定数 a の組合せで得られるパターンは全て冗長でない共通パターンである（性質1）。従って、冗長でない1変数共通パターンの数は、$\sum_{i=0}^{k} {}_{2^k}C_{2^i} = \sum_{i=0}^{k} {}_nC_{2^i} = {}_nC_1 + {}_nC_2 + {}_nC_4 + \cdots + {}_nC_{n/2} + {}_nC_n$ とおり存在する。

〔性質3〕(1) 集合Sの1変数共通パターンpの中に変数 x が i 回出現しているとする。この時 i $(0 < i \leq |s_1|)$ は、整数集合 $\{|s_2| - |s_1|, |s_3| - |s_2|, \cdots, |s_m| - |s_{m-1}|\}$ の公約数でなければならない。また、文字列 s_j の中で x に置き換えられた部分列の長さを l_j とした時 $l_{j+1} = l_j + (|s_{j+1}| - |s_j|)/i$ が成り立つ。

〔定義4〕 2つのパターン P_1, P_2 について、$P \in \Sigma^*$ かつ $P \leq P_1$ かつ $P \leq P_2$ かつ $|P| \leq |s_1|$ であり、$q \leq P_1$ かつ $q \leq P_2$ かつ $P \leq q$ となる様なパターン q が存在しない時、P を P_1 と P_2 の積と言い、$P_1 \wedge P_2$ と書く。上記の条件を満たす P が存在しない時、$P_1 \wedge P_2$ は定義されない。

〔例3〕 $S = \{aabb, aababbb\}$ とする。$P_1 = aax$, $P_2 = axb$ とした時、$P_1 \wedge P_2 = aaxb$ である。また $P_1 = aax$

$P_3 = x\,bb$ とした時，$P_1 \wedge P_3$ は定義されない。

〔性質4〕　2つの1変数共通パターン P_1，P_2 において，$P = P_1 \wedge P_2$ が定義されている時，$L(P) = L(P_1) \cap L(P_2)$ となる。

〔性質5〕　P_1，P_2 を与えられた文字列集合 S の1変数共通パターンとする。この時，$P_1 \wedge P_2$ が定義されており，$P_1 \wedge P_2$ が1変数パターンであれば，$P_1 \wedge P_2$ は S の1変数共通パターンである。逆に，$P_1 \wedge P_2$ が1変数共通パターンであれば，P_1 P_2 のいずれか1方は冗長な共通パターンである。

4．1変数共通パターンを求めるアルゴリズム

本節では与えられた文字列集合 $S = \{s_1, s_2, \cdots, s_m\}$ の冗長でない1変数共通パターンを求める効率良いアルゴリズムを示す。Angluin は性質3を利用して，$n = |s_m|$ とした時，$O(mn^5)$ の計算時間を要するアルゴリズムを示した[1]。ここでは $Max(O(mn^2), O(mn_1^3))$ の計算時間を要するアルゴリズムを示す（$n_1 = |s_1|$）。ただし，S の冗長でない1変数共通パターンの数は n に関して指数関数的に増大する場合があるので，ここでは全ての解を示す構造のみを求めるアルゴリズムを与える。

〔定義5〕　与えられた文字列 $s = s(1)s(2)\cdots s(n)$ $(s(i) \in \Sigma,$ $1 \leq i \leq n)$ に対して $s' = s\lambda$ $(\lambda \notin \Sigma)$ を考える。s' の i-位置指示

系列 (i-th position identifier) W_i とは次の条件を満足する α' の部分系列である。

① $W_i = \alpha'(i)\alpha'(i+1)\cdots\alpha'(j)$ $(i \leq j \leq n+1)$ である。

② $i=j$ の時，$W_i = \alpha'(i)$ は α' の中にただ1回だけ現れる記号である。$i<j$ の時，W_i は α' の中にただ1回だけ現れる部分系列でありしかも，$\alpha'(i)\alpha'(i+1)\cdots\alpha'(j-1)$ は α' の中に少なくとも2回以上現れる部分系列である。

〔定義6〕 文字列 $\alpha = \alpha(1)\alpha(2)\cdots\alpha(n)$ に対する prefix 木とは次の条件を満足する木である。$\alpha' = \alpha\lambda$ とする。

① prefix 木は $n+1$ 個の葉節点を持ち，各葉節点には $1 \sim n+1$ の整数が1意にラベル付けされている。

② 各枝には，$\Sigma \cup \{\lambda\}$ の要素がラベル付けされている。

③ prefix 木の根節点よりラベル i の葉節点へ到る道上のラベルを順に連接して得られる系列は，α' の i-位置指示系列である。

〔例4〕 $\Sigma = \{a, b\}$，$\omega = aababa$ の prefix 木を図1に示す。例えば，$abab$ は ω の2-位置指示系列である。

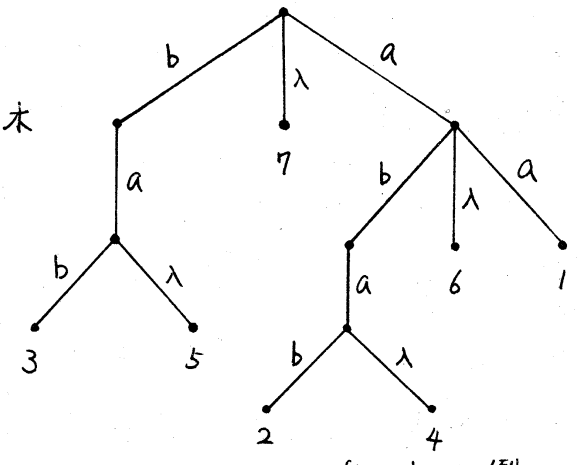

図1 prefix 木の例

〔性質6〕 文字列Λの中で繰り返し現れる文字列は全て，Λのprefix木に表現されている[2]。

1変数共通パターンを求める為に，各文字列の中で繰り返し現れる文字列を求める必要がある。次に，prefix木を用いて全ての繰し返し現れる文字列を効率良く求めるためのアルゴリズムを示す。Prefix木の根節点と葉節点以外の節点を内部節点と呼ぶことにする。

〔アルゴリズム1〕繰り返し系列を求めるアルゴリズム

(1)文字列Λに対するprefix木（Tと書く）を構成する。

(2)内部節点Pを根節点とするTの部分木の全ての葉節点のラベルの集合をPに対応づける。

(3)節点Pの深さを$d(P)$とし，ステップ(2)によって得られた集合を$\{i_1, i_2, \cdots, i_k\}$とした時，$\{\Lambda(i_1)\Lambda(i_1+1)\cdots\Lambda(i_1+d(P)-1), \cdots, \Lambda(i_k)\Lambda(i_k+1)\cdots\Lambda(i_k+d(P)-1)\}$が求める長さ$d(P)$の繰り返し系列である。

アルゴリズム1 終り

〔定理1〕$|\Lambda|=n$とした時，アルゴリズム1は$O(n^2)$の計算時間を必要とする。

《証》ステップ(1)は$O(n)$の時間を必要とする[2]。全ての内部節点Pに対して，ステップ(2)はTをエンドオーダーに1度だけトラバースすることによって実行される。Tの節点数は$O(n)$であることが知られている。従って，Tのトラバース

には $O(n)$ の時間を要する。一方各内部節点は、$O(n)$ の大きさのラベル集合を持つ。従ってアルゴリズム1は全体として $O(n^2)$ の計算時間を必要とする。　　Q.E.D.

次に本節で示すアルゴリズムで利用するデータ構造を定義する。定義8で示すパターン網は prefix木を用いて得られた繰り返し系列を利用して簡単に構成できる。

〔定義7〕 大きさ $k \times l$ の網とは、$(k+1) \times (l+1)$ 個の節点を持ち、次の条件を満たす有向グラフである。

(1)各節点は2値組 (i,j) $(0 \leq i \leq k, 0 \leq j \leq l)$ で表現され、枝は (i,j) より $(i,j+1)$ 或いは $(i+1,j)$ に向けてのみ存在する。

(2)各枝には $\Sigma \cup X$ の要素がラベル付けされている。

〔定義8〕 文字列 w に対する (i,j) パターン網とは大きさ $i \times j$ の網であり、定数 i 個、変数 j 個より成る $\{w\}$ の冗長でない1変数共通パターンが全て、点 $(0,0)$ より点 (i,j) に到る有向道で表現されているものを言う。

〔性質7〕 w の (i,j) パターン網において $(0,0)$ より (i,j) への有向道が存在するための必要十分条件は、w の中に長さ $(|w|-i)/j$ の文字列が少なくとも j 回繰り返し出現していることである。但し、文字列が重なってはならない。

〔例5〕 $w = aababa$ に対する $(2,2)$ パターン網を図2に示す。ここで変数 x_1 は文字列 "ab"、x_2 は文字列 "ba" を示

している。wの(2,2)パターン網を求める為に，図1のprefix木を用い，長さ2で少なくとも2回繰り返しwの中に現れる文字列を求めれば良い。

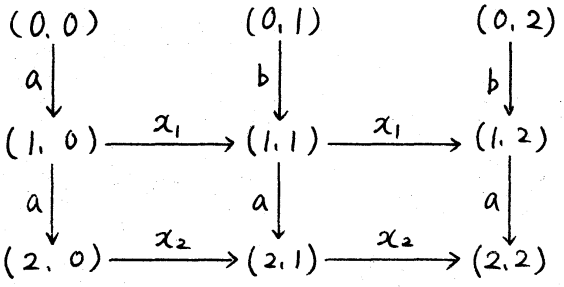

図2　(2,2)パターン網の例

次に文字列wの(i,j)パターン網を構成するためのアルゴリズムを示す。ただし，(|w|−i)/jは整数とする。

〔アルゴリズム2〕(i,j)パターン網を求めるアルゴリズム

(1)文字列wにアルゴリズム1を適用し，wのprefix木の全ての内部節点Pについて，対応したラベル集合を求める。

(2) (i+1)×(j+1)個の節点を作る。n=(|w|−i)/jとする。

(3) ∀k,l (0≤k≤i−1, 0≤l≤j)について，節点(k,l)から節点(k+1,l)へ枝を作り，記号w(nl+k+1)をラベル付けする。u=0とする。

(4) wのprefix木の深さnの新しい内部節点Pを求める。Pがなければアルゴリズムは終了。

(5) u=u+1とする。Pに対応したラベル集合(ステップ(1))を考える。そのラベル集合の各要素i_tについてステップ(6)を行なう。

(6) $i_t=1+k+nl$を満足する全ての整数k,lに対し，点(k,l)

より点 $(k, l+1)$ へ枝をつくり，ラベル x_u をつける。

(7) ステップ (4) へゆく。　　アルゴリズム2終り

〔定理2〕　アルゴリズム2は，$|w| = n$ とした時，$O(n^2) + O(ij)$ の計算時間を必要とする。

《証》 ステップ(1)は，定理1より $O(n^2)$ の計算時間を必要とする。ステップ(2)及びステップ(3)はそれぞれ，$(i+1)\cdot(j+1)$，$i\cdot(j+1)$ の計算量を要する。ステップ(4)は prefix 木をトラバースすればよいので $O(n)$ の時間で処理できる。ステップ(5)～(6)によって，パターン網の右向きの枝が作られるが，各枝のラベルは一意に決定される。従って，(5)～(6)で必要な計算時間は作られる枝数に比例し，枝数は高々 $(i+1)\cdot j$ 本である。従って全体として，$O(n^2) + O(3(i+1)(j+1))$ の計算量を要する。 Q.E.D.

文字列 $w \in S$ に対する (i,j) パターン網は，i, j を決めればその構造が一意に決まる。また S の1変数共通パターンの内、定数 i 個，x (変数) j 個より成るパターンがあれば，定義より，それは w の (i,j) パターン網に表現されている。この事実により，S の各要素に対する (i,j) パターン網の共通部分を求めれば，S の1変数共通パターン網が求まる。公式なアルゴリズムを次に示す。

〔アルゴリズム3〕 1変数共通パターンを求めるアルゴリズム

(1) 各 $w \in S$ に対して，アルゴリズム1を適用する。

(2) 整数集合 $\{|s_2|-|s_1|, |s_3|-|s_2|, \cdots, |s_m|-|s_{m-1}|\}$ の公約数の集合（以後CDと書く）を求める。

(3) for $j=1$ until $|s_1|$ do begin

if $j \in CD$ then begin

(4) for $h=1$ until $\lfloor |s_1|/j \rfloor$ do begin

$i \leftarrow |s_1| - h \cdot j$

s_1 について，(i,j) パターン網 $N_1(i,j)$ を作る。

for $k=2$ until m do begin

s_k の (i,j) パターン網 $N_k(i,j)$ と $N_1(i,j)$ の共通部分を改めて $N_1(i,j)$ とする。

end

(5) $N_1(i,j)$ において，点 $(0,0)$ より点 (i,j) へ到る有向道が存在すれば，$N_1(i,j)$ は解を与える構造である。

(6) end

(7) end

アルゴリズム3 終り

〔定理3〕 アルゴリズム3は，$\mathrm{Max}(O(mn^2), O(mn_1^3))$ の計算時間を必要とする。但し $n_1=|s_1|$，$n=|s_m|$ とする。

《証》 ステップ(1)は定理1より，高々 $O(mn^2)$ の計算時間を要する。ループ(3)～(7)及びループ(4)～(6)は，それぞれ最悪の場合，n_1 回及び $\lfloor n_1/i \rfloor$ 回繰り返される。ループ(4)～(6)の中で，各文字列 s_k ごとに (i,j) パターン網が構成される。但し実際には，s_1 の (i,j) パターン網上での操作が行なわれるのみである。ステップ(5)は，$(i+1)(j+1)$ の時間で処理できる。

従って、ステップ(3)～(7)に対して次の評価式を得る。

$$\sum_{j=1}^{n_1}\sum_{h=1}^{\lfloor m/j\rfloor}\left\{(i+1)(j+1)+i(j+1)+(i+1)j+\sum_{k=2}^{m}\left(i(j+1)+(i+1)j\right)+(i+1)(j+1)\right\}$$

$$\leq \sum_{j=1}^{n_1}\sum_{h=1}^{\lfloor m/j\rfloor}\left\{2(i+1)(j+1)+m(2ij+i+j)\right\} < (m+2)\sum_{j}\sum_{h}(i+1)(j+1)$$

$$=(m+2)\sum_{j}\sum_{h}(j+1)(n_1-hj+1)=O(mn_1^3).$$

従って、全体として定理3の結果を得る。　　Q.E.D.

〔例6〕 $S=\{aababa, abbabb aa\}$ の1変数共通パターンの内、定数2、変数2のものを考える。

$s_1=aababa$ の(2,2)パターン網は図2に示すとおりである。$s_2=abb\,abbaa$ の(2,2)パターン網を図3に示す。2つの(2,2)パターン網の共通部分を図4に示す。

図4より、Sの1変数共通パターンの1つは、$axxa$ であることがわかる。また、定数2、変数2個より成る1変数共通パターンはこれ以外にない。

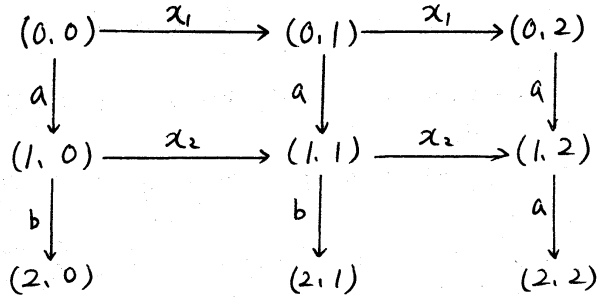

図3　$s_2=abb\,abbaa$ の(2,2)パターン網

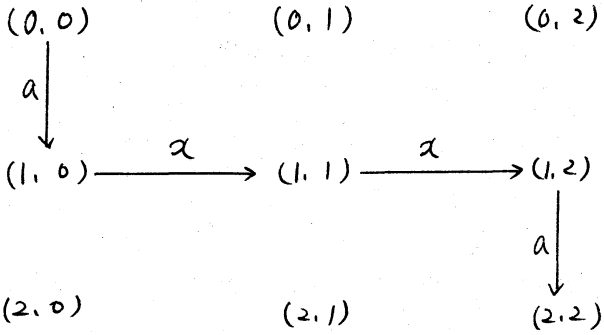

図4.　図2と図3の(2,2)パターン網の共通部分

5. 多変数共通パターンについて

任意の種類の変数を許した場合に、与えられた文字列集合Sに対し、Sの冗長でない共通パターンの内で長さ最大のものをみつけることはNP-hardであることが既に知られている[1]。この結果からみても、一般の共通パターンを求めることは非常に難かしいと言える。1変数の場合に成立していた性質1も、2変数以上では成立しない。また、1変数共通パターンの場合には、変数と定数の個数を決めれば与えられた文字列に対して、変数が表現する文字列の長さは一意的に決定されたが、2変数以上の場合にはこれも成り立たない。また、性質2の様に実行時間の減少に寄与する性質も判明していない。そこで本節では、変数の種類に制限を加えて、共通パターンの問題を考える。すなわち、Σ上っ文字列から文字列への同型写像の集合を$\{f_i\}$とした時、$\{f_i(x)\}$なる変数の集合のみを考えることにする。

同型写像の有限集合を$\{f_1, f_2, \cdots, f_k\}$（但し、$f_1$は恒等写像とする）とした時、変数として$f_i(x)$ $(1 \leq i \leq k)$を用いた共通パターンを$O(kmn^3)$で求めるアルゴリズムを示すことができる。まず、例を用いて我々のアルゴリズムを説明する。

〔例7〕 $\Sigma = \{a, b\}$とし、変数としてx及びx^Rを許した共通パターンを考える。与えられた文字列集合を$S = \{aababa,$

abbaaabb } とする。Sの共通パターンの内、定数2個、変数2個より成るパターンを求める為に、(2,2) パターン網を構成する。その為に、s_1 = aababa の長さ2の部分列 $s_{1,2}$ と $s_{1,2}^R$ の出現位置を全て求める。また、s_2 = abbaaabb の長さ3の部分列 $s_{2,3}$ と $s_{2,3}^R$ の出現位置を全て求める。例えば $s_{2,3}$ = abb とした時、$s_{2,3}$ と $s_{2,3}^R$ の s_2 における出現位置を全て求める為に、図5に示す様な有限オートマトンを考えればよい[3][4]。ここで状態0が初期状態、状態3,6が最終状態である。s_1, s_2 の (2,2) パターン網を各々、図6,7に示す。図6において、x_1, x_2 は各々長さ2の文字列 ab, ba を示している。図7において、x_1, x_2, x_3, x_4, x_5 は各々 abb, bba, baa, aaa, aab を示している。

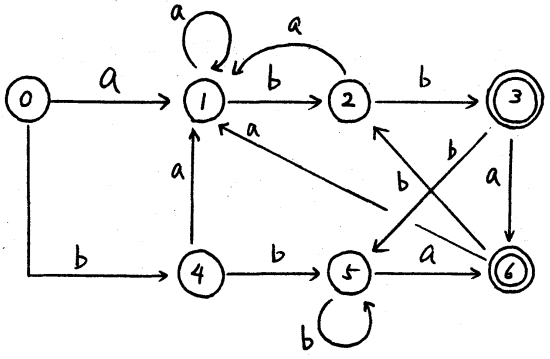

図5 {abb, bba} を受理する有限オートマトン

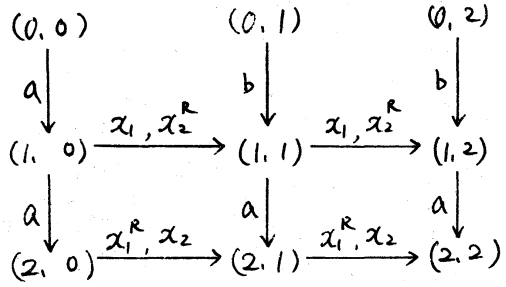

図6 s_1 の (2,2) パターン網

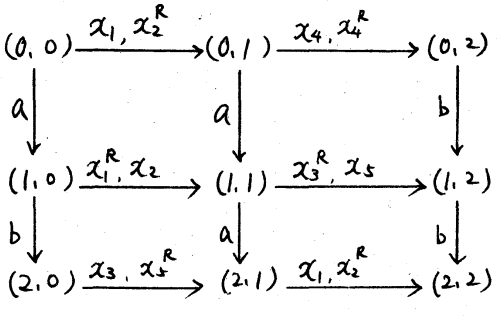

図7 s_2 の (2,2) パターン網

図6と図7の共通部分を考えることにより，定数2，変数2個のSの共通パターン $ax\,ax^R$, $ax^R\,ax$ を得る。

公式なアルゴリズムを次に示す。

〔アルゴリズム4〕

(1) for $t=1$ until m do begin

(2) for $p=1$ until $|s_t|$ do begin

文字列 s_t の長さ p の部分列を $s_{t,p}$ とする。
全ての $s_{t,p}$ について以下の処理を行なう。

(3) 文字列集合 $\{f_1(s_{t,p}), f_2(s_{t,p}), \cdots, f_k(s_{t,p})\}$ を受理する有限オートマトンを構成する。
その有限オートマトンに s_t を入力することにより，文字列 $f_j(s_{t,p})$ $(1 \le j \le k)$ の出現位置を全て求める。

(4) end

(5) end

(6) アルゴリズム3のステップ(2)以下を適用する。

アルゴリズム4終り

〔定理4〕 アルゴリズム4の計算時間は，$O(kmn^3)$ である。但し，$n=|s_m|$，k は同型写像の数である。

(証) 文字列 s_t の長さ p の部分列は高々 $|s_t|-p+1$ 種類存在する。ステップ(3)は $O(kp+|s_t|)$ の時間で計算できる[3][4]。故にステップ(1)～(5)は全体として，$O(kmn^3)$ の計算量を必要とする。一方，ステップ(6)では，アルゴリズム3の場合と同様の計算時間を要するが，パターン網の右向きの枝には，高々

k種のラベルが付けられる可能性がある。従ってステップ(1)は$O(kmn^3)$の計算時間を要する。 Q.E.D.

6. あとがき　本稿では1変数共通パターンを求める有効なアルゴリズムと、変数をx及びxに同型写像を施した変数に制限した場合の多変数共通パターンを求めるアルゴリズムを示した。本稿で述べたパターンの形を拡張して、より一般的なパターンを定義することができる。しかしながら、この様なパターンまで含めて一般的な共通パターンを見つけ出すことは非常に難かしい問題である。

謝辞

御討論頂いた京都産業大学・岩間一雄先生に感謝します。また日頃より貴重な御助言を賜わる京都大学情報工学教室・矢島脩三先生に感謝致します。

〔参考文献〕

[1] Angluin, D., "Finding Patterns Common to a Set of Strings", 11th ACM Symposium on Theory of Computing, pp. 130-141, 1979.

[2] Weiner, P., "Linear Pattern Matching Algorithms", 14th Annual Symposium on Switching and Automata Theory, pp. 1-11, 1973.

[3] Aho, A.V. and Corasick, M.J., "Efficient String Matching: An Aid to Bibliographic Search", CACM, vol. 8, no. 6, pp. 333-340, June 1975.

[4] Kambayashi, Y., Nakatsu, N. and Yajima, S., "Hierarchical String Pattern Matching Using Dynamic Pattern Matching Machines", The Proceedings of IEEE COMPSAC'79, pp. 813-818, Nov. 1979.